Using Condor
An Introduction

Condor Week 2005

Condor Project
Computer Sciences Department
University of Wisconsin-Madison
condor-admin@cs.wisc.edu
http://www.cs.wisc.edu/condor
Tutorial Outline

The Story of Frieda, the Scientist

- Using Condor to manage jobs
- Using Condor to manage resources
- Condor Architecture and Mechanisms
- Condor on the Grid
  - Flocking
  - Condor-G

Stop me if you have any questions!
Meet Frieda.

She is a scientist. But she has a big problem.
Frieda’s Application ...

Run a Parameter Sweep of $F(x,y,z)$ for 20 values of $x$, 10 values of $y$ and 3 values of $z$ ($20 \times 10 \times 3 = 600$ combinations)

- $F$ takes on the average 6 hours to compute on a “typical” workstation (total = 3600 hours)
- $F$ requires a “moderate” (128MB) amount of memory
- $F$ performs “moderate” I/O - ($x,y,z$) is 5 MB and $F(x,y,z)$ is 50 MB

http://www.cs.wisc.edu/condor
I have 600 simulations to run.

Where can I get help?
As if by magic, a genie appears from a lamp, and says, “Install a Personal Condor!”
Getting Condor

› Available as a free download from http://www.cs.wisc.edu/condor

› Download Condor for your operating system
  • Available for most UNIX (including Linux) platforms
  • Also for Windows NT / XP
Condor Releases

- Stable / Developer Releases
  - Naming scheme similar to the Linux Kernel...
  - **Major.minor.release**
    - Minor is even (a.b.c): Stable
      - Examples: 6.4.3, 6.6.8, 6.6.9
      - Very stable, mostly bug fixes
    - Minor is odd (a.b.c): Developer
      - New features, may have some bugs
      - Examples: 6.5.5, 6.7.5, 6.7.6
Frieda Installs a “Personal Condor” on her machine…

› What do we mean by a “Personal” Condor?
   • Condor on your own workstation, no root access required, no system administrator intervention needed

› After installation, Frieda submits her jobs to her Personal Condor…

http://www.cs.wisc.edu/condor
Frieda’s Condor Pool

F(3,4,5)

600 Condor jobs

personal Condor

Frieda’s workstation

http://www.cs.wisc.edu/condor
Personal Condor?!

What’s the benefit of a Condor “Pool” with just one user and one machine?
Your Personal Condor will ... 

› ... keep an eye on your jobs and will keep you posted on their progress
› ... implement your policy on the execution order of the jobs
› ... keep a log of your job activities
› ... add fault tolerance to your jobs
› ... implement your policy on when the jobs can run on your workstation
Getting Started: Submitting Jobs to Condor

› Choosing a “Universe” for your job
  • Just use VANILLA for now
› Make your job “batch-ready”
› Creating a submit description file
› Run condor_submit on your submit description file
Making your job batch-ready

Must be able to run in the background: no interactive input, windows, GUI, etc.

Can still use `STDIN`, `STDOUT`, and `STDERR` (the keyboard and the screen), but files are used for these instead of the actual devices

Organize data files
Creating a Submit Description File

› A plain ASCII text file
› Condor does not care about file extensions
› Tells Condor about your job:
   • Which executable, universe, input, output and error files to use, command-line arguments, environment variables, any special requirements or preferences (more on this later)
› Can describe many jobs at once (a “cluster”), each with different input, arguments, output, etc.
Simple Submit Description File

# Simple condor_submit input file
# (Lines beginning with # are comments)
# NOTE: the words on the left side are not
#       case sensitive, but filenames are!
Universe       = vanilla
Executable     = my_job
Queue

http://www.cs.wisc.edu/condor
Running condor_submit

› You give `condor_submit` the name of the submit file you have created:
  • `condor_submit my_job.submit`

› `condor_submit` parses the submit file, checks for it errors, and creates a "ClassAd" that describes your job(s)
  • ClassAds: Condor’s internal data representation
    • Similar to classified ads (as the name implies)
    • Represent an object & it’s attributes
    • Can also describe what an object matches with
The Job Queue

- `condor_submit` sends your job’s ClassAd(s) to the schedd
  - Manages the local job queue
  - Stores the job in the job queue
    - Atomic operation, two-phase commit
    - “Like money in the bank”

- View the queue with `condor_q`
Running condor_submit

% condor_submit my_job.submit
Submitting job(s).
1 job(s) submitted to cluster 1.

% condor_q

-- Submitter: perdita.cs.wisc.edu : <128.105.165.34:1027> :
   ID      OWNER            SUBMITTED     RUN_TIME ST PRI SIZE CMD
   1.0    frieda  6/16 06:52   0+00:00:00 I  0   0.0  my_job

1 jobs; 1 idle, 0 running, 0 held

%
More information about jobs

- Controlled by submit file settings

- Condor sends you email about events
  - Turn it off: Notification = Never
  - Only on errors: Notification = Error

- Condor creates a log file (user log)
  - “The Life Story of a Job”
  - Shows all events in the life of a job
  - Always have a log file
  - To turn it on: Log = filename

http://www.cs.wisc.edu/condor
Sample Condor User Log

000 (0001.000.000) 05/25 19:10:03 Job submitted from host: <128.105.146.14:1816>
...

001 (0001.000.000) 05/25 19:12:17 Job executing on host: <128.105.146.14:1026>
...

005 (0001.000.000) 05/25 19:13:06 Job terminated.

(1) Normal termination (return value 0)

  Usr 0 00:00:37, Sys 0 00:00:00  - Run Remote Usage
  Usr 0 00:00:00, Sys 0 00:00:05  - Run Local Usage
  Usr 0 00:00:37, Sys 0 00:00:00  - Total Remote Usage
  Usr 0 00:00:00, Sys 0 00:00:05  - Total Local Usage

  9624  - Run Bytes Sent By Job
  7146159  - Run Bytes Received By Job
  9624  - Total Bytes Sent By Job
  7146159  - Total Bytes Received By Job

...
Another Submit Description File

# Example condor_submit input file
# (Lines beginning with # are comments)
# NOTE: the words on the left side are not
#       case sensitive, but filenames are!
Universe  = vanilla
Executable = /home/frieda/condor/my_job.condor
Log       = my_job.log
Input     = my_job.stdin
Output    = my_job.stdout
Error     = my_job.stderr
Arguments = -arg1 -arg2
InitialDir = /home/frieda/condor/run_1
Queue
“Clusters” and “Processes”

› If your submit file describes multiple jobs, we call this a “cluster”
› Each cluster has a unique “cluster number”
› Each job in a cluster is called a “process”
  • Process numbers always start at zero
› A Condor “Job ID” is the cluster number, a period, and the process number (“20.1”)
  • A cluster can have only one process (“21.0”)
Example Submit Description File for a Cluster

# Example submit description file that defines a
# cluster of 2 jobs with separate working directories
Universe   = vanilla
Executable = my_job
log        = my_job.log
Arguments  = -arg1 -arg2
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
InitialDir = run_0
Queue
· Becomes job 2.0
InitialDir = run_1
Queue
· Becomes job 2.1
Submitting The Job

% condor_submit my_job.submit-file

Submitting job(s).
2 job(s) submitted to cluster 2.

% condor_q

-- Submitter: perdita.cs.wisc.edu : <128.105.165.34:1027> :

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>SUBMITTED</th>
<th>RUN_TIME</th>
<th>ST</th>
<th>PRI</th>
<th>SIZE</th>
<th>CMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>frieda</td>
<td>4/15 06:52</td>
<td>0:02:11</td>
<td>R</td>
<td>0</td>
<td>0.0</td>
<td>my_job</td>
</tr>
<tr>
<td>2.0</td>
<td>frieda</td>
<td>4/15 06:56</td>
<td>0:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>my_job</td>
</tr>
<tr>
<td>2.1</td>
<td>frieda</td>
<td>4/15 06:56</td>
<td>0:00:00</td>
<td>I</td>
<td>0</td>
<td>0.0</td>
<td>my_job</td>
</tr>
</tbody>
</table>

3 jobs; 2 idle, 1 running, 0 held

%
Submit Description File for a **BIG Cluster of Jobs**

- The initial directory for each job can be specified as run_$(Process), and instead of submitting a single job, we use “Queue 600” to submit 600 jobs at once.
- The $(Process) macro will be expanded to the process number for each job in the cluster (0 - 599), so we’ll have “run_0”, “run_1”, … “run_599” directories.
- All the input/output files will be in different directories!
Submit Description File for a BIG Cluster of Jobs

# Example condor_submit input file that defines
# a cluster of 600 jobs with different directories
Universe   = vanilla
Executable = my_job
Log        = my_job.log
Arguments  = -arg1 -arg2
Input      = my_job.stdin
Output     = my_job.stdout
Error      = my_job.stderr
InitialDir = run_$(Process)  # run_0 ... run_599
Queue 600  # Becomes job 3.0 ... 3.599

http://www.cs.wisc.edu/condor
Using condor_rm

- If you want to remove a job from the Condor queue, you use `condor_rm`
- You can only remove jobs that you own (you can’t run `condor_rm` on someone else’s jobs unless you are root)
- You can give specific job ID’s (cluster or cluster.proc), or you can remove all of your jobs with the “-d” option.
  - `condor_rm 21.1` Removes a single job
  - `condor_rm 21` Removes a whole cluster
Frieda's Condor Pool

Frieda can still only run one job at a time, however.
Good News

The Boss says Frieda can add her co-workers’ desktop machines into her Condor pool as well… but only if they can also submit jobs.

http://www.cs.wisc.edu/condor
Adding nodes

- Frieda installs Condor on the desktop machines, and configures them with her machine as the central manager
- These are “non-dedicated” nodes, meaning that they can’t always run Condor jobs
Frieda’s Condor Pool

Now, Frieda and her co-workers can run multiple jobs at a time so their work completes sooner.

http://www.cs.wisc.edu/condor
% condor_status

<table>
<thead>
<tr>
<th>Name</th>
<th>OpSys</th>
<th>Arch</th>
<th>State</th>
<th>Activity</th>
<th>LoadAv</th>
<th>Mem</th>
<th>ActvtyTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>haha.cs.wisc.</td>
<td>IRIX65</td>
<td>SGI</td>
<td>Unclaimed</td>
<td>Idle</td>
<td>0.198</td>
<td>192</td>
<td>0+00:00:04</td>
</tr>
<tr>
<td>antipholus.cs</td>
<td>LINUX</td>
<td>INTEL</td>
<td>Unclaimed</td>
<td>Idle</td>
<td>0.020</td>
<td>511</td>
<td>0+02:28:42</td>
</tr>
<tr>
<td>coral.cs.wisc.</td>
<td>LINUX</td>
<td>INTEL</td>
<td>Claimed</td>
<td>Busy</td>
<td>0.990</td>
<td>511</td>
<td>0+01:27:21</td>
</tr>
<tr>
<td>doc.cs.wisc.e</td>
<td>LINUX</td>
<td>INTEL</td>
<td>Unclaimed</td>
<td>Idle</td>
<td>0.260</td>
<td>511</td>
<td>0+00:20:04</td>
</tr>
<tr>
<td>dsonokwa.cs.w</td>
<td>LINUX</td>
<td>INTEL</td>
<td>Claimed</td>
<td>Busy</td>
<td>0.810</td>
<td>511</td>
<td>0+00:01:45</td>
</tr>
<tr>
<td>ferdinand.cs.</td>
<td>LINUX</td>
<td>INTEL</td>
<td>Claimed</td>
<td>Suspended</td>
<td>1.130</td>
<td>511</td>
<td>0+00:00:55</td>
</tr>
<tr>
<td>vm1@pinguino.</td>
<td>LINUX</td>
<td>INTEL</td>
<td>Unclaimed</td>
<td>Idle</td>
<td>0.000</td>
<td>255</td>
<td>0+01:03:28</td>
</tr>
<tr>
<td>vm2@pinguino.</td>
<td>LINUX</td>
<td>INTEL</td>
<td>Unclaimed</td>
<td>Idle</td>
<td>0.190</td>
<td>255</td>
<td>0+01:03:29</td>
</tr>
</tbody>
</table>
How can my jobs access their data files?
Access to Data in Condor

- Use Shared Filesystem if available
- No shared filesystem?
  - *Condor can transfer files*
    - Can automatically send back changed files
    - Atomic transfer of multiple files
    - Can be encrypted over the wire
  - Remote I/O Socket
  - Standard Universe can use remote system calls (more on this later)
Condor File Transfer

- ShouldTransferFiles = YES
  - Always transfer files to execution site
- ShouldTransferFiles = NO
  - Rely on a shared filesystem
- ShouldTransferFiles = IF_NEEDED
  - Will automatically transfer the files if the submit and execute machine are not in the same FileSystemDomain

Universe = vanilla
Executable = my_job
Log = my_job.log
ShouldTransferFiles = IF_NEEDED
Transfer_input_files = dataset$(Process), common.data
Transfer_output_files = TheAnswer.dat
Queue 600
We Need More

- Condor is managing and running our jobs, but:
  - Our CPU requirements are greater than our resources
  - Jobs get vacated when people use their workstations

http://www.cs.wisc.edu/condor
Happy Day! Frieda’s organization purchased a Dedicated Cluster!

- Frieda Installs Condor on all the dedicated Cluster nodes
- Frieda also adds a dedicated central manager
- She configures her entire pool with this new host as the central manager...
Frieda’s Condor Pool

With the additional resources, Frieda and her co-workers can get their jobs completed even faster.
What Condor Daemons are running on my machine, and what do they do?
condor_master

- Starts up all other Condor daemons
- If there are any problems and a daemon exits, it restarts the daemon and sends email to the administrator
- Acts as the server for many Condor remote administration commands:
  - `condor_reconfig`, `condor_restart`, `condor_off`, `condor_on`, `condor_config_val`, etc.
Condor Daemon Layout

Personal Condor / Central Manager

Master

- startd
- schedd
- negotiator
- collector

\[\text{process spawns} = \text{process spawned}\]
condor_collector

- Only on the Central Manager
- “Defines” your Condor Pool
- One Collector per pool
- Collects information from all other Condor daemons in the pool
  - “Directory Service” / Database for a Condor pool
- Each daemon sends a periodic update called a “ClassAd” to the collector
- Services queries for information:
  - Queries from other Condor daemons
  - Queries from users (condor_status)

http://www.cs.wisc.edu/condor
Layout of the Condor Pool

\[\text{Central Manager}\]

- 

\[\text{Master}\]

\[\text{Collector}\]

\[\text{\ldots\ldots\ldots\ldots\ = Process Spawned}\]

\[\text{\longrightarrow = ClassAd}\]

\[\text{Communication Pathway}\]

http://www.cs.wisc.edu/condor
condor_startd

› Represents a machine to the Condor system
› Responsible for starting, suspending, and stopping jobs
› Enforces the wishes of the machine owner (the owner’s “policy”... more on this in the admin tutorial)
› Only on “execute” nodes

http://www.cs.wisc.edu/condor
Layout of the Condor Pool

Diagram:
- Central Manager:
  - Master
  - Collector

- Cluster Node:
  - Master
  - startd

Arrow types:
- ➔ = Process Spawned
- ➔ = ClassAd Communication Pathway

http://www.cs.wisc.edu/condor
condor_schedd

- Only on “submit nodes” (hosts that you can submit jobs from)
- Maintains the persistent queue of jobs
- Responsible for contacting available machines and sending them jobs
- Services user commands which manipulate the job queue:
  - `condor_submit, condor_rm, condor_q, condor_hold, condor_release, condor_prio, ...`
Layout of the Condor Pool

- Process Spawned
- ClassAd Communication Pathway

Central Manager
- Master
- Collector

Cluster Node
- Master
- startd

Cluster Node
- Master
- startd

Desktop
- Master
- startd
- schedd

Desktop
- Master
- startd
- schedd

http://www.cs.wisc.edu/condor
condor_negotiator

- Only on Central Manager
- Only one negotiator per pool
- Performs “matchmaking” in Condor
- Gets information from the collector about all available machines and all idle jobs
- Tries to match jobs with machines that will serve them
- Both the job and the machine must satisfy each other’s requirements
Layout of the Condor Pool

= Process Spawned

= ClassAd Communication Pathway

Central Manager

Master

negotiator

schedd

Collector

Cluster Node

Master

startd

Cluster Node

Master

startd

Cluster Node

Master

startd

Cluster Node

Master

startd

Desktop

Master

startd

schedd

Desktop

Master

startd

schedd

Desktop

Master

startd

schedd

Desktop

Master

startd

schedd

http://www.cs.wisc.edu/condor
Some of the machines in the Pool do not have enough memory or scratch disk space to run my job!
Specify Requirements!

- An expression (syntax similar to C or Java)
- Must evaluate to True for a match to be made

```
Universe = vanilla
Executable = my_job
Log = my_job.log
InitialDir = run_$(Process)
Requirements = Memory >= 256 && Disk > 10000
Queue 600
```
Specify Rank!

› All matches which meet the requirements can be sorted by preference with a Rank expression.

› Higher the Rank, the better the match

Universe   = vanilla
Executable = my_job
Log        = my_job.log
Arguments  = -arg1 -arg2
InitialDir = run_$(Process)
Requirements = Memory >= 256 && Disk > 10000
Rank = (KFLOPS*10000) + Memory
Queue 600
We've seen how Condor can:

... keeps an eye on your jobs and will keep you posted on their progress

... implements your policy on the execution order of the jobs

... keeps a log of your job activities
My jobs run for 20 days...

- What happens when they get pre-empted?
- How can I add fault tolerance to my jobs?
Condor’s **Standard Universe**

to the rescue!

- Condor can support various combinations of features/environments in different “Universes”
- Different Universes provide different functionality for your job:
  - **Vanilla** – Run any Serial Job
  - **Scheduler** – Plug in a meta-scheduler
  - **Standard** – Support for transparent process checkpoint and restart

http://www.cs.wisc.edu/condor
Process Checkpointing

Condor’s Process Checkpointing mechanism saves the entire state of a process into a checkpoint file:
- Memory, CPU, I/O, etc.

The process can then be restarted from right where it left off.

Typically no changes to your job’s source code needed – however, your job must be relinked with Condor’s Standard Universe support library.

http://www.cs.wisc.edu/condor
Relinking Your Job for Standard Universe

To do this, just place "condor_compile" in front of the command you normally use to link your job:

% condor_compile gcc -o myjob myjob.c
- OR -
% condor_compile f77 -o myjob filea.f fileb.f
- OR -
% condor_compile make -f MyMakefile
Limitations of the Standard Universe

- Condor’s checkpointing is not at the kernel level. Thus in the Standard Universe the job may not:
  - Fork()
  - Use kernel threads
  - Use some forms of IPC, such as pipes and shared memory
- Many typical scientific jobs are OK
When will Condor checkpoint your job?

› Periodically, if desired
  • For fault tolerance
› When your job is preempted by a higher priority job
› When your job is vacated because the execution machine becomes busy
› When you explicitly run `condor_checkpoint, condor_vacate, condor_off` or `condor_restart` command

http://www.cs.wisc.edu/condor
Remote I/O Socket

› Job can request that the condor_starter process on the execute machine create a Remote I/O Socket

› Used for online access of file on submit machine - without Standard Universe.
  • Use in Vanilla, Java, ...

› Libraries provided for Java and for C, e.g.:
  Java: FileInputStream -> ChirpInputStream
  C: open() -> chirp_open()
Remote System Calls

› I/O System calls are trapped and sent back to submit machine
› Allows Transparent Migration Across Administrative Domains
  • Checkpoint on machine A, restart on B
› No Source Code changes required
› Language Independent
› Opportunities for Application Steering
  • Example: Condor tells customer process “how” to open files

http://www.cs.wisc.edu/condor
Job Startup

- Schedd
- Shadow
- Submit
- Startd
- Customer Job
- Condor Syscall Lib
- Starter

http://www.cs.wisc.edu/condor
```bash
c01(69)% condor_q -io

-- Submitter: c01.cs.wisc.edu : <128.105.146.101:2996> : c01.cs.wisc.edu

<table>
<thead>
<tr>
<th>ID</th>
<th>OWNER</th>
<th>READ</th>
<th>WRITE</th>
<th>SEEK</th>
<th>XPUT</th>
<th>BUFSIZE</th>
<th>BLKSIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>72.3</td>
<td>edayton</td>
<td>[ no i/o data collected yet ]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>72.5</td>
<td>edayton</td>
<td>6.8 MB</td>
<td>0.0 B</td>
<td>0</td>
<td>104.0 KB/s</td>
<td>512.0 KB</td>
<td>32.0 KB</td>
</tr>
<tr>
<td>73.0</td>
<td>edayton</td>
<td>6.4 MB</td>
<td>0.0 B</td>
<td>0</td>
<td>140.3 KB/s</td>
<td>512.0 KB</td>
<td>32.0 KB</td>
</tr>
<tr>
<td>73.2</td>
<td>edayton</td>
<td>6.8 MB</td>
<td>0.0 B</td>
<td>0</td>
<td>112.4 KB/s</td>
<td>512.0 KB</td>
<td>32.0 KB</td>
</tr>
<tr>
<td>73.4</td>
<td>edayton</td>
<td>6.8 MB</td>
<td>0.0 B</td>
<td>0</td>
<td>139.3 KB/s</td>
<td>512.0 KB</td>
<td>32.0 KB</td>
</tr>
<tr>
<td>73.5</td>
<td>edayton</td>
<td>6.8 MB</td>
<td>0.0 B</td>
<td>0</td>
<td>139.3 KB/s</td>
<td>512.0 KB</td>
<td>32.0 KB</td>
</tr>
<tr>
<td>73.7</td>
<td>edayton</td>
<td>[ no i/o data collected yet ]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0 jobs; 0 idle, 0 running, 0 held
```
Connecting Condors

› Frieda knows people with their own Condor pools, and gets permission to use their computing resources...

› How can Condor help her do this?
Connect Condors with Flocking

› Frieda configures her Condor pool to “flock” to her friend’s pool.
› Flocking is a Condor-specific technology.
Frieda’s Condor Pool

600 Condor jobs

Condor Pool

Friendly Condor Pool

http://www.cs.wisc.edu/condor
Frieda meets The Grid

› Frieda also has access to Globus resources she wants to use
  • She has certificates and access to Globus gatekeepers at remote institutions

› But Frieda wants Condor’s queue management features for her Globus jobs!

› She installs Condor-G so she can submit “Globus Universe” jobs to Condor
Condor-G: Globus + Condor

**Globus**
- middleware deployed across entire Grid
- remote access to computational resources
- dependable, robust data transfer

**Condor**
- job scheduling across multiple resources
- strong fault tolerance with checkpointing and migration
- layered over Globus as “personal batch system” for the Grid

http://www.cs.wisc.edu/condor
Condor-G Installation

› Install Condor from the Condor web site
  • Condor-G is “included” as Globus Universe

  -- OR --

› Install from NMI

  -- OR --

› Install from VDT
Frieda Submits a Globus Universe Job

› In her submit description file, Frieda specifies:
  • Universe = Globus
  • Which Globus Gatekeeper to use
  • Optional: Location of file containing your Globus certificate

```
universe = globus
globusscheduler = beak.cs.wisc.edu/jobmanager
executable = progname
queue
```
How Condor-G Works

Personal Condor

Schedd

Globus Resource

LSF
How Condor-G Works

Personal Condor

Globus Resource

Schedd

600 Globus jobs

http://www.cs.wisc.edu/condor
How Condor-G Works

Personal Condor

Schedd

GridManager

Globus Resource

LSF

http://www.cs.wisc.edu/condor
How Condor-G Works

Personal Condor

Schedd

GridManager

Globus Resource

JobManager

LSF

http://www.cs.wisc.edu/condor
How Condor-G Works

Personal Condor
- Schedd
- GridManager

Globus Resource
- JobManager
- LSF
- User Job

http://www.cs.wisc.edu/condor

600 Globus jobs
Globus Universe Concerns

What about Fault Tolerance?
  • Local Crashes
    • What if the submit machine goes down?
  • Network Outages
    • What if the connection to the remote Globus jobmanager is lost?
  • Remote Crashes
    • What if the remote Globus jobmanager crashes?
    • What if the remote machine goes down?
Changes to the Globus JobManager for Fault Tolerance

› Ability to restart a JobManager
› Enhanced two-phase commit submit protocol
Globus Universe Fault-Tolerance: Submit-side Failures

- All relevant state for each submitted job is stored persistently in the Condor job queue.
- This persistent information allows the Condor GridManager upon restart to read the state information and reconnect to JobManagers that were running at the time of the crash.
- If a JobManager fails to respond...
Globus Universe Fault-Tolerance: Credential Management

- Authentication in Globus is done with limited-lifetime X509 proxies
- Proxy may expire before jobs finish executing
- Condor can put jobs on hold and email user to refresh proxy
My jobs have have dependencies...

Can Condor help solve my dependency problems?
Frieda learns DAGMan

» Directed Acyclic Graph Manager

» DAGMan allows you to specify the dependencies between your Condor jobs, so it can manage them automatically for you.

» (e.g., “Don’t run job “B” until job “A” has completed successfully.”)
What is a DAG?

› A DAG is the data structure used by DAGMan to represent these dependencies.

› Each job is a “node” in the DAG.

› Each node can have any number of “parent” or “children” nodes - as long as there are no loops!
Defining a DAG

› A DAG is defined by a *.dag* file, listing each of its nodes and their dependencies:

```plaintext
# diamond.dag
Job A a.sub
Job B b.sub
Job C c.sub
Job D d.sub
Parent A Child B C
Parent B C Child D
```

› each node will run the Condor job specified by its accompanying *Condor submit file*
Submitting a DAG

› To start your DAG, just run `condor_submit_dag` with your .dag file, and Condor will start a personal DAGMan daemon which to begin running your jobs:

```bash
% condor_submit_dag diamond.dag
```

› `condor_submit_dag` submits a Scheduler Universe Job with DAGMan as the executable.

› Thus the DAGMan daemon itself runs as a Condor job, so you don’t have to baby-sit it.
Running a DAG

- **DAGMan** acts as a “meta-scheduler”, managing the submission of your jobs to **Condor** based on the DAG dependencies.
Running a DAG (cont'd)

DAGMan holds & submits jobs to the Condor queue at the appropriate times.
Running a DAG (cont'd)

> In case of a job failure, DAGMan continues until it can no longer make progress, and then creates a "rescue" file with the current state of the DAG.
Once the failed job is ready to be re-run, the rescue file can be used to restore the prior state of the DAG.
Recovering a DAG (cont'd)

› Once that job completes, DAGMan will continue the DAG as if the failure never happened.
Finishing a DAG

- Once the DAG is complete, the DAGMan job itself is finished, and exits.
Additional DAGMan Features

› Provides other handy features for job management...

• nodes can have **PRE & POST** scripts
• failed nodes can be automatically re-tried a configurable number of times
• job submission can be “throttled”
General User Commands

- condor_status
  View Pool Status
- condor_q
  View Job Queue
- condor_submit
  Submit new Jobs
- condor_rm
  Remove Jobs
- condor_prio
  Intra-User Prios
- condor_history
  Completed Job Info
- condor_submit_dag
  Specify Dependencies
- condor_checkpoint
  Force a checkpoint
- condor_compile
  Link Condor library
**Administrator Commands**

- `condor_vacate` Leave a machine now
- `condor_on` Start Condor
- `condor_off` Stop Condor
- `condor_reconfig` Reconfig on-the-fly
- `condor_config_val` View/set config
- `condor_userprio` User Priorities
- `condor_stats` View detailed usage accounting stats

http://www.cs.wisc.edu/condor
Condor Job Universes

- Serial Jobs
  - Vanilla Universe
  - Standard Universe
- Scheduler Universe
- Parallel Jobs
  - MPI Universe
  - PVM Universe
- Java Universe

http://www.cs.wisc.edu/condor
Java Universe Job

```
universe = java
executable = Main.class
jar_files = MyLibrary.jar
input = infile
output = outfile
arguments = Main 1 2 3
queue
```

condor_submit
Why not use Vanilla Universe for Java jobs?

Java Universe provides more than just inserting “java” at the start of the execute line

- Knows which machines have a JVM installed
- Knows the location, version, and performance of JVM on each machine
- Provides more information about Java job completion than just JVM exit code
  - Program runs in a Java wrapper, allowing Condor to report Java exceptions, etc.
### Java support, cont.

**condor_status -java**

<table>
<thead>
<tr>
<th>Name</th>
<th>JavaVendor</th>
<th>Ver</th>
<th>State</th>
<th>Activity</th>
<th>LoadAv</th>
<th>Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td>aish.cs.wisc.</td>
<td>Sun Microsysy</td>
<td>1.2.2</td>
<td>Owner</td>
<td>Idle</td>
<td>0.000</td>
<td>249</td>
</tr>
<tr>
<td>anfrom.cs.wis</td>
<td>Sun Microsysy</td>
<td>1.2.2</td>
<td>Owner</td>
<td>Idle</td>
<td>0.030</td>
<td>249</td>
</tr>
<tr>
<td>babe.cs.wisc.</td>
<td>Sun Microsysy</td>
<td>1.2.2</td>
<td>Claimed</td>
<td>Busy</td>
<td>1.120</td>
<td>123</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Condor]

[http://www.cs.wisc.edu/condor]
Job Policy Expressions

- User can supply job policy expressions in the submit file.
- Can be used to describe a successful run.

```plaintext
on_exit_remove = <expression>
on_exit_hold = <expression>
periodic_remove = <expression>
periodic_hold = <expression>
```
Job Policy Examples

› Do not remove if exits with a signal:
  on_exit_remove = ExitBySignal == False

› Place on hold if exits with nonzero status or ran for less than an hour:
  on_exit_hold = ((ExitBySignal==False) && (ExitSignal != 0)) || ((ServerStartTime - JobStartDate) < 3600)

› Place on hold if job has spent more than 50% of its time suspended:
  periodic_hold = CumulativeSuspensionTime > (RemoteWallClockTime / 2.0)
CondorView Usage Graph


Total Condor
Total Idle
Total Owner

Configure...  Zoom In  Zoom Out  Reset  About

http://www.cs.wisc.edu/condor
But Frieda Wants More…

› She wants to run standard universe jobs on Globus-managed resources
  • For matchmaking and dynamic scheduling of jobs
    • Note: Condor-G will now do matchmaking!
  • For job checkpointing and migration
  • For remote system calls
Solution: Condor GlideIn

- Frieda can use the Globus Universe to run Condor daemons on Globus resources
- When the resources run these GlideIn jobs, they will temporarily join her Condor Pool
- She can then submit Standard, Vanilla, PVM, or MPI Universe jobs and they will be matched and run on the Globus resources
600 Condor jobs

Condor Pool

glide-in jobs

Friendly Condor Pool

Condor

PBS

LSF

Globus Grid
How It Works

Personal Condor

- Schedd
- Collector

Globus Resource

- LSF

http://www.cs.wisc.edu/condor
How It Works

Personal Condor

600 Condor jobs

Schedd

Collector

GlideIn jobs

Globus Resource

LSF

http://www.cs.wisc.edu/condor
How It Works

Personal Condor

- Schedd
- GridManager
- Collector

Globus Resource

- LSF

http://www.cs.wisc.edu/condor
How It Works

Personal Condor

- Schedd
- GridManager
- Collector

Globus Resource

- JobManager
- LSF

600 Condor jobs

GlideIn jobs

http://www.cs.wisc.edu/condor
How It Works

Personal Condor
- Schedd
- GridManager
- Collector

Globus Resource
- JobManager
- LSF
- Startd

600 Condor jobs

GlideIn jobs

http://www.cs.wisc.edu/condor
How It Works

Personal Condor

Schedd

GridManager

Collector

GlideIn jobs

Globus Resource

JobManager

LSF

Startd

http://www.cs.wisc.edu/condor
How It Works

Personal Condor

- Schedd
- GridManager
- Collector

Globus Resource

- JobManager
- LSF
- Startd
- User Job

600 Condor jobs

http://www.cs.wisc.edu/condor
Job Submission Machine

- End User Requests
- Condor-G Scheduler
- Condor-G GridManager
- GASS Server
- Condor-G Collector
- Condor Shadow Process for Job X

Job Execution Site

- Globus Daemons + Local Site Scheduler
- [See Figure 1]

Job

- Condor Daemons
- Transfer Job X
- Redirected System Call Data
- Condor System Call Trapping & Checkpoint Library

http://www.cs.wisc.edu/condor
GlideIn Concerns

› What if a Globus resource kills my GlideIn job?
  • That resource will disappear from your pool and your jobs will be rescheduled on other machines
  • Standard universe jobs will resume from their last checkpoint like usual

› What if all my jobs are completed before a GlideIn job runs?
  • If a GlideIn Condor daemon is not matched with a job in 10 minutes, it terminates, freeing the resource
A Common Question

My Personal Condor is flocking with a bunch of Solaris machines, and also doing a GlideIn to a Silicon Graphics O2K. I do not want to statically partition my jobs.

Solution: In your submit file, specify:

```
Executable = myjob.$$\text{(OpSys)}.$$\text{(Arch)}
```

The "\$\$(xxx)" notation is replaced with attributes from the machine ClassAd which was matched with your job.
In Review

With Condor Frieda can...

• ... manage her compute job workload
• ... access local machines
• ... access remote Condor Pools via flocking
• ... access remote compute resources on the Grid via Globus Universe jobs
• ... carve out her own personal Condor Pool from the Grid with GlideIn technology

http://www.cs.wisc.edu/condor
Thank you!

Check us out on the Web:
http://www.condorproject.org

Email:
condor-admin@cs.wisc.edu